HAL
archives-ouvertes.fr

Nonlinear corrector for RANS equations

Loic Frazza, Adrien Loseille, Frédéric Alauzet, Alain Dervieux

▶ To cite this version:

Loic Frazza, Adrien Loseille, Frédéric Alauzet, Alain Dervieux. Nonlinear corrector for RANS equations. 2018 Fluid Dynamics Conference, AIAA AVIATION Forum, Jun 2018, Atlanta, United States. <hal-01962171>

HAL Id: hal-01962171
https://hal.inria.fr/hal-01962171

Submitted on 20 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Nonlinear corrector for RANS equations

Loïc Frazza*, Adrien Loseille† Frédéric Alauzet‡ Alain Dervieux §

Sorbonne Universités, UPMC, 4 place Jussieu 75252 Paris cedex 05, France

Gamma3 Team, Inria Saclay, 91120 Palaiseau, France

The scope of this paper is to present a nonlinear error estimation and correction for Navier-Stokes and RANS equations. This correction is obtained by deducing a source term from the evaluation of the residual of the solution interpolated on the $h/2$ mesh. To avoid the generation of the $h/2$ mesh (which is prohibitive for realistic applications), the residual at each vertex is computed by local refinement only in the neighborhood of the considered vertex. It successfully improves solution predictions and yields a sharp estimate of the numerical error.

I. Introduction

The numerical simulation of engineering problems involves a discrete solution which is alleged to converge toward the continuous solution of the problem as the mesh-size decreases. As the exact analytical solution of the problem is sought, the difference between the numerical and the analytical solution can be seen as a noise. In an engineering context, this noise can have disastrous consequences on the numerical prediction which can, in turn, lead to actual accidents or misconceptions.

It is thus essential to reduce and estimate this error. This is usually done by leading a grid convergence study, where the error is estimated with the difference between two solutions computed on two successives grids. However, this approach requires an additional finer grid (thus expensive) to appropriately estimate the error. Moreover, it requires the expertise of an engineer. In a mesh adaptation context multiple successive grids are automatically generated following a sensor which estimates the error related to the local mesh size. It does not requires any mesh prescription, but it needs a relatively accurate estimation of the numerical error to be efficient and to stop when the mesh and the solution are converged.

Recently, different adjoint based mesh adaptation strategies[1–6] have been developed in order to optimize the mesh with respect to a specific functional output. These techniques uses the adjoint solution to estimate the error committed on the output and minimize it with respect to the mesh size. However, these approaches are based on a linearization of the equations and can thus be inefficient for strongly nonlinear problems like RANS equations. This is why we proposed a nonlinear equivalent of these approaches in order to efficiently compute a correction in the global solution and in the output functional (lift, drag, heat transfer, sound, ...).

*PhD student, Sorbonne Universités, UPMC Univ Paris 06, Gamma3 Team

†Researcher, Gamma3 Team

‡Researcher, Gamma3 Team

§Researcher, Gamma3 Team

II. Adjoint based error estimation and correction

A. Problem setup and Numerical discretization

1. Navier-Stokes RANS equations

The compressible Navier-Stokes equations for mass, momentum and energy conservation read:

$$\left\{\begin{array}{rcl} \dfrac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) &=& 0\,, \\ \dfrac{\partial(\rho\mathbf{u})}{\partial t} + \nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u}) + \nabla p &=& \nabla\cdot\mathcal{T}\,, \\ \dfrac{\partial(\rho E)}{\partial t} + \nabla\cdot((\rho E + p)\mathbf{u}) &=& \nabla\cdot(\mathcal{T}\cdot\mathbf{u}) + \nabla\cdot(\lambda\nabla T)\,, \end{array}\right. \tag{1}$$

where ρ denotes the density (kg/m^3), $\mathbf{u}$ the velocity (m/s), E the total energy per mass ($m^2.s^{-2}$), p the pressure (N/m^2), T the temperature (K), μ the laminar dynamic viscosity ($kg/(m.s)$) and λ the laminar conductivity. $\mathcal{T}$ the laminar stress tensor:

$$\mathcal{T} = (\mu+\mu_t)\left[(\nabla\otimes\mathbf{u} + {}^t\nabla\otimes\mathbf{u}) - \frac{2}{3}\nabla.\,\mathbf{u}\,\mathbb{I}\right],$$

where (in 3D) $\mathbf{u} = (u, v, w)$ and

$$\nabla.\,\mathbf{u}\,\mathbb{I} = \begin{pmatrix} u_x+v_y+w_z & 0 & 0 \\ 0 & u_x+v_y+w_z & 0 \\ 0 & 0 & u_x+v_y+w_z \end{pmatrix},$$

where $u_x = \frac{\partial u}{\partial x}$, $u_y = \frac{\partial u}{\partial y}$, $u_z = \frac{\partial u}{\partial z}$ (idem for v and w).

The variation of nondimensionalized laminar dynamic viscosity and conductivity coefficients μ and λ as a function of the dimensional temperature T is defined by Sutherland's law:

$$\mu \;=\; \mu_\infty\left(\frac{T}{T_\infty}\right)^{\frac{3}{2}}\left(\frac{T_\infty+\mathrm{Su}}{T+\mathrm{Su}}\right) \quad\text{and}\quad \lambda \;=\; \lambda_\infty\left(\frac{T}{T_\infty}\right)^{\frac{3}{2}}\left(\frac{T_\infty+\mathrm{Su}}{T+\mathrm{Su}}\right),$$

where $\mathrm{Su} = 110$ is the Sutherland constant and the index ∞ denotes reference quantities. The relation linking μ and λ is expressed from the Prandtl laminar number:

$$\mathrm{Pr} = \frac{\mu c_p}{\lambda} \quad\text{with}\quad \mathrm{Pr} = 0.72 \quad\text{for (dry) air}\,,$$

where C_p is the specific heat at constant pressure.

2. Turbulence modeling

According to the standard approach to turbulence modeling based upon the Boussinesq hypothesis, the turbulence is modeled with an eddy viscosity μ_t, which is added to the laminar (or dynamic) viscosity μ. The dynamic viscosity is usually taken to be a function of the temperature, whereas μ_t is obtained using a turbulence model. Here we choose the Spalart-Allmaras one equation turbulence model[7] given by the following equation:

$$\frac{\partial\tilde{\nu}}{\partial t} + \mathbf{u}\cdot\nabla\tilde{\nu} = c_{b1}[1-f_{t2}]\tilde{S}\tilde{\nu} - \left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + \frac{1}{\sigma}\left[\nabla\cdot((\nu+\tilde{\nu})\nabla\tilde{\nu}) + c_{b2}\|\nabla\tilde{\nu}\|^2\right] + f_{t1}\Delta\mathbf{u}^2\,, \tag{2}$$

where $\tilde{\nu}$ is the turbulent kinematic viscosity and all the constants are defined below. In the standard model the trip term is being left out, *i.e.,* $f_{t1} = 0$. Moreover, some implementations also ignore the f_{t2} term as it is argued that if the trip is not included, then f_{t2} is not necessary.[8] In `Wolf`, this simplified version has been considered and we prefer to write it under the following form, which is more appropriate for its discretization

with the finite element/finite volume method. Indeed, Equation (2) can be decomposed into the following terms:

$$\frac{\partial \rho\tilde{\nu}}{\partial t} + \underbrace{\mathbf{u}\cdot\nabla\rho\tilde{\nu}}_{convection} = \underbrace{c_{b1}\tilde{S}\rho\tilde{\nu}}_{production} \underbrace{- c_{w1} f_w \rho \left(\frac{\tilde{\nu}}{d}\right)^2}_{destruction} + \underbrace{\frac{\rho}{\sigma}\nabla\cdot((\nu+\tilde{\nu})\nabla\tilde{\nu})}_{dissipation} + \underbrace{\frac{c_{b2}\rho}{\sigma}\|\nabla\tilde{\nu}\|^2}_{diffusion} .$$

Notice that this is not a conservative model. If a conservative form of the Spalart-Allmaras is foreseen, we have to consider the variation proposed by Catris and Aupoix.[9] The turbulent eddy viscosity is computed from:

$$\mu_t = \rho\tilde{\nu} f_{v1} ,$$

where

$$f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \quad \text{and} \quad \chi = \frac{\tilde{\nu}}{\nu} \quad \text{with} \quad \nu = \frac{\mu}{\rho} .$$

Additional definitions are given by the following equations:

$$f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}} \quad \text{and} \quad \tilde{S} = \Omega + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2} \quad \text{where} \quad \Omega = \|\nabla\times\mathbf{u}\| .$$

d is the distance to nearest wall which is computed for each vertex at the beginning of the simulation. The set of closure constants for the model is given by

$$\sigma = \frac{2}{3} , \qquad c_{b1} = 0.1355 , \quad c_{b2} = 0.622 , \quad \kappa = 0.41 ,$$
$$c_{w1} = \frac{c_{b1}}{\kappa} + \frac{1+c_{b2}}{\sigma} , \quad c_{w2} = 0.3 , \qquad c_{w3} = 2 , \qquad c_{v1} = 7.1 .$$

Finally, the function f_w is computed as:

$$f_w = g\left(\frac{1+c_{w3}^6}{g^6 + c_{w3}^6}\right)^{1/6} \quad \text{with} \quad g = r + c_{w2}\left(r^6 - r\right) \quad \text{and} \quad r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2} , 10\right) .$$

3. Vector form of the RANS system

We write the RANS system in the following (more compact) vector form:

$$W_t + F_1(W)_x + F_2(W)_y + F_3(W)_z = S_1(W)_x + S_2(W)_y + S_3(W)_z + Q(W) ,$$

where $S_i(W)_a = \dfrac{\partial S_i(W)}{\partial a}$ $(i = 1,2,3, a = x,y,z)$ (idem for F). W is the nondimensionalized conservative variables vector:

$$W = (\rho, \rho u, \rho v, \rho w, \rho E, \rho\tilde{\nu})^T .$$

$F(W) = (F_1(W), F_2(W), F_3(W))$ are the convective (Euler) flux functions:

$$\begin{aligned} F_1(W) &= \left(\rho u,\ \rho u^2 + p,\ \rho uv,\ \rho uw,\ u(\rho E + p), \rho u\tilde{\nu}\right)^T , \\ F_2(W) &= \left(\rho v,\ \rho uv,\ \rho v^2 + p,\ \rho vw,\ v(\rho E + p), \rho v\tilde{\nu}\right)^T , \\ F_3(W) &= \left(\rho w,\ \rho uw,\ \rho vw,\ \rho w^2 + p,\ w(\rho E + p), \rho w\tilde{\nu}\right)^T . \end{aligned} \tag{3}$$

$S(W) = (S_1(W), S_2(W), S_3(W))$ are the laminar viscous fluxes:

$$\begin{aligned} S_1(W) &= \left(0,\ \mathcal{T}_{xx},\ \mathcal{T}_{xy},\ \mathcal{T}_{xz},\ u\mathcal{T}_{xx} + v\mathcal{T}_{xy} + w\mathcal{T}_{xz} + \lambda T_x, \frac{\rho}{\sigma}(\nu+\tilde{\nu})\tilde{\nu}_x\right)^T , \\ S_2(W) &= \left(0,\ \mathcal{T}_{xy},\ \mathcal{T}_{yy},\ \mathcal{T}_{yz},\ u\mathcal{T}_{xy} + v\mathcal{T}_{yy} + w\mathcal{T}_{yz} + \lambda T_y, \frac{\rho}{\sigma}(\nu+\tilde{\nu})\tilde{\nu}_y\right)^T , \\ S_3(W) &= \left(0,\ \mathcal{T}_{xz},\ \mathcal{T}_{yz},\ \mathcal{T}_{zz},\ u\mathcal{T}_{xz} + v\mathcal{T}_{yz} + w\mathcal{T}_{zz} + \lambda T_z, \frac{\rho}{\sigma}(\nu+\tilde{\nu})\tilde{\nu}_z\right)^T , \end{aligned} \tag{4}$$

where $\mathcal{T}_{ij}$ are the components of laminar stress tensor $\mathcal{T} = (\mu + \mu_t)\left[(\nabla \otimes \mathbf{u} + {}^t\nabla \otimes \mathbf{u}) - \frac{2}{3}\nabla.\,\mathbf{u}\,\mathbb{I}\right]$:

$$\mathcal{T}_{xx} = (\mu+\mu_t)\frac{2}{3}\left(2u_x - v_y - w_z\right), \quad \mathcal{T}_{xy} = (\mu+\mu_t)\left(u_y + v_x\right), \quad \mathcal{T}_{xz} = (\mu+\mu_t)\left(u_z + w_x\right), \quad ...$$

$Q(W)$ are the source terms, *i.e.* the diffusion, production and destruction terms from the Spalart-Allmaras turbulence model:

$$Q(W) = (0,\ 0,\ 0,\ 0,\ 0, \frac{c_{b2}\rho}{\sigma}\|\nabla\tilde{\nu}\|^2 + \rho c_{b1}\tilde{S}\tilde{\nu} + c_{w1}f_w\rho\left(\frac{\tilde{\nu}}{d}\right)^2)^T. \tag{5}$$

Note that $Q = 0$ in the case of the laminar Navier-Stokes equations, unless additional source terms are added (to take into account gravity, for instance).

4. Spatial discretization

The spatial discretization of the fluid equations (1) and (2) is based on a vertex-centered finite element/finite volume formulation on unstructured meshes. The equations are integrated on each finite volume cell C_i (using the Green formula):

$$|C_i|\frac{dW_i}{dt} + \mathbf{F}_i = \mathbf{S}_i + \mathbf{Q}_i\,, \tag{6}$$

where W_i is the mean value of the solution W on cell C_i, $\mathbf{F}_i$, $\mathbf{S}_i$ and $\mathbf{Q}_i$ are respectively the numerical convective, viscous and source flux terms:

$$\mathbf{F}_i = \int_{\partial C_i} F(W_i)\cdot\mathbf{n}_i \mathrm{d}\gamma\,, \qquad \mathbf{S}_i = \int_{\partial C_i} S(W_i)\cdot\mathbf{n}_i\,\mathrm{d}\gamma\,, \qquad \mathbf{Q}_i = \int_{C_i} Q(W_i)\,\mathrm{d}\Omega\,,$$

where $\mathbf{n}_i$ is the outer normal to the finite volume cell surface ∂C_i, and F, S and Q are respectively the convective, viscous and source terms flux functions as defined previously in Relations (3), (4) and (5). The numerical scheme combines a HLLC approximate Riemann solver[10] to compute the convective fluxes and the Galerkin centered method to evaluate the viscous terms. Second order space accuracy is achieved through a piecewise linear extrapolation based on the Monotonic Upwind Scheme for Conservation Law (MUSCL) procedure[11] which uses a particular edge-based formulation with upwind elements.

HLLC approximate Riemann solver. The idea of the HLLC flow solver is to consider locally a simplified Riemann problem with two intermediate states depending on the local left and right states. The simplified solution to the Riemann problem consists of a contact wave with a velocity S_M and two acoustic waves, which may be either shocks or expansion fans. The acoustic waves have the smallest and the largest velocities (S_i and S_j, respectively) of all the waves present in the exact solution. If $S_i > 0$ then the flow is supersonic from left to right and the upwind flux is simply defined from $F(W_i)$ where W_i is the state to the left of the discontinuity. Similarly, if $S_j < 0$ then the flow is supersonic from right to left and the flux is defined from $F(W_j)$ where W_j is the state to the right of the discontinuity. In the more difficult subsonic case when $S_i < 0 < S_j$ we have to calculate $F(W_i^*)$ or $F(W_j^*)$. Consequently, the HLLC flux is given by:

$$\Phi_{ij}^{HLLC}(W_i, W_j, \mathbf{n}_{ij}) = \begin{cases} F(W_i)\cdot\mathbf{n}_{ij} & \text{if } S_i > 0 \\ F(W_i^\star)\cdot\mathbf{n}_{ij} & \text{if } S_i \le 0 < S_M \\ F(W_j^\star)\cdot\mathbf{n}_{ij} & \text{if } S_M \le 0 \le S_j \\ F(W_j)\cdot\mathbf{n}_{ij} & \text{if } S_j < 0 \end{cases}.$$

W_i^* and W_j^* are evaluated as follows. We denote by $\eta = \mathbf{u}\cdot\mathbf{n}$. Assuming that $\eta^* = \eta_i^* = \eta_j^* = S_M$, the following evaluations are proposed[10] (the subscripts $_i$ and $_j$ are omitted for clarity):

$$W^* = \frac{1}{S - S_M}\begin{pmatrix} \rho\,(S-\eta) \\ \rho\mathbf{u}\,(S-\eta) + (p^* - p)\mathbf{n} \\ \rho E\,(S-\eta) + p^* S_M - p\eta \end{pmatrix} \quad \text{where} \quad p^* = \rho\,(S-\eta)(S_M - \eta) + p\,.$$

A key feature of this solver is in the definition of the three waves velocity. For the contact wave we consider:

$$S_M = \frac{\rho_j \eta_j (S_J - \eta_j) - \rho_i \eta_i (S_I - \eta_i) + p_i - p_j}{\rho_j (S_J - \eta_j) - \rho_i (S_I - \eta_i)} ,$$

and the acoustic wave speeds based on the Roe average:

$$S_I = \min(\eta_i - c_i, \tilde{\eta} - \tilde{c}) \quad \text{and} \quad S_J = \max(\eta_j + c_j, \tilde{\eta} + \tilde{c}) .$$

With such waves velocities, the approximate HLLC Riemann solver has the following properties. It automatically (i) satisfies the entropy inequality, (ii) resolves isolated contacts exactly, (iii) resolves isolated shocks exactly, and (iv) preserves positivity.

2ND-ORDER ACCURATE VERSION. The MUSCL type reconstruction method has been designed to increase the order of accuracy of the scheme.[11] The idea is to use extrapolated values W_{ij} and W_{ji} instead of W_i and W_j at the interface ∂C_{ij} to evaluate the flux with the approximate Riemann solver. Note that, in the implementation, the primitive variables $(\rho, \mathbf{u}, p)$ are extrapolated to guarantee the positivity of the density and the pressure, then the conservative variables are reconstructed from these values. Thus, the gradients of the primitive variables are evaluated. However, in the following, we still denote by W the primitive variables vector. The numerical flux becomes:

$$\Phi_{ij} = \Phi_{ij}^{HLLC}(W_{ij}, W_{ji}, \mathbf{n}_{ij}) ,$$

where W_{ij} and W_{ji} are linearly extrapolated as:

$$W_{ij} \;=\; W_i + \frac{1}{2} (\nabla W)_i \cdot \overrightarrow{P_i P_j} \qquad \text{and} \qquad W_{ji} \;=\; W_j + \frac{1}{2} (\nabla W)_j \cdot \overrightarrow{P_j P_i} .$$

In contrast to the original MUSCL approach, the approximate "slopes" $(\nabla W)_{ij}$ and $(\nabla W)_{ji}$ are defined for any edge and obtained using a combination of centered, upwind and nodal gradients.

The centered gradient, which is related to edge $P_i P_j$, is implicitly defined along edge $P_i P_j$ by the relation:

$$(\nabla W)_{ij}^C \cdot \overrightarrow{P_i P_j} = W_j - W_i \qquad \text{and} \qquad (\nabla W)_{ij}^C \cdot \overrightarrow{P_j P_i} = W_i - W_j .$$

Upwind and downwind gradients, which are also related to edge $P_i P_j$, are computed according to the definition of upwind and downwind tetrahedra of edge $P_i P_j$. These tetrahedra are respectively denoted K_{ij} and K_{ji}. K_{ij} (resp. K_{ji}) is the unique tetrahedron of the ball of P_i (resp. P_j) the opposite face of which is crossed by the line defined by the edge $P_i P_j$, see Figure 1. Upwind and downwind gradients are then defined

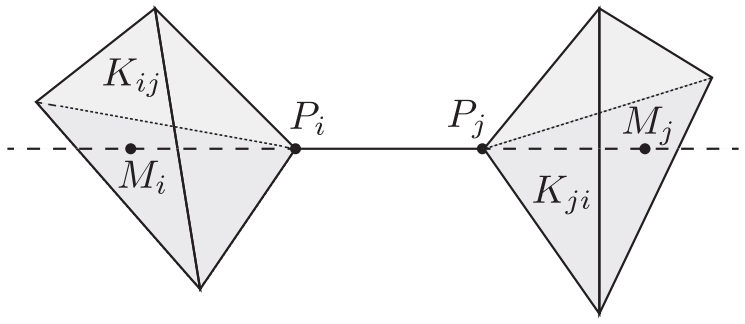

Figure 1. Downwind K_{ij} and upwind K_{ji} tetrahedra associated with edge $P_i P_j$.

for vertices P_i and P_j as:

$$(\nabla W)_{ij}^U = (\nabla W)|_{K_{ij}} \quad \text{and} \quad (\nabla W)_{ij}^D = (\nabla W)|_{K_{ji}} .$$

where $(\nabla W)|_K = \sum_{P \in K} W_P \nabla \phi_P|_K$ is the P_1-Galerkin gradient on tetrahedron K. Parametrized gradients are built by introducing the β-scheme:

$$\begin{aligned} (\nabla W)_i \cdot \overrightarrow{P_i P_j} &= (1-\beta)(\nabla W)_{ij}^C \cdot \overrightarrow{P_i P_j} + \beta \, (\nabla W)_{ij}^U \cdot \overrightarrow{P_i P_j} \\ (\nabla W)_j \cdot \overrightarrow{P_j P_i} &= (1-\beta)(\nabla W)_{ij}^C \cdot \overrightarrow{P_j P_i} + \beta \, (\nabla W)_{ij}^D \cdot \overrightarrow{P_j P_i} , \end{aligned}$$

where $\beta \in [0,1]$ is a parameter controlling the amount of upwinding. For instance, the scheme is centered for $\beta = 0$ and fully upwind for $\beta = 1$.

FOURTH-ORDER NUMERICAL DISSIPATION: V4-SCHEME. The most accurate β-scheme is obtained for $\beta = 1/3$. Indeed, it can be demonstrated that this scheme is third-order for the two-dimensional linear advection on structured triangular meshes. On unstructured meshes, a second-order scheme with a fourth-order numerical dissipation is obtained. These high-order gradients are given by:

$$\begin{aligned}(\nabla W)_i^{V4} \cdot \overrightarrow{P_iP_j} &= \frac{2}{3}(\nabla W)_{ij}^{C} \cdot \overrightarrow{P_iP_j} + \frac{1}{3}(\nabla W)_{ij}^{U} \cdot \overrightarrow{P_iP_j} \\ (\nabla W)_j^{V4} \cdot \overrightarrow{P_jP_i} &= \frac{2}{3}(\nabla W)_{ij}^{C} \cdot \overrightarrow{P_jP_i} + \frac{1}{3}(\nabla W)_{ij}^{D} \cdot \overrightarrow{P_jP_i}.\end{aligned}$$

The MUSCL schemes are not monotone and can be a source of spurious oscillations especially in the vicinity of discontinuities.[12] These oscillations can affect the accuracy of the final solution or simply end the computation because (for instance) of negative pressures. A widely used technique for addressing this issue is to guarantee the TVD property in 1D[13] or the LED property in 2D/3D of the scheme, which ensures that the extrapolated values W_{ij} and W_{ji} are not invalid. To guarantee the TVD or the LED properties, limiting functions are coupled with the previous high-order gradient evaluations. The gradient is substituted by a limited gradient denoted $(\nabla W)_{ij}^{lim}$. The choice of the limiting function is crucial as it directly affects the convergence of the simulation. In this work, we use the Piperno limiters[14,15] which is expressed in a factorized form,

$$(\nabla W)^{Lim} = \nabla W^C \; \psi_{PI}\left(\frac{\nabla W^C}{\nabla W^{V4}}\right),$$

with

$$\psi_{PI}(R) = (\frac{1}{3} + \frac{2}{3}R) \begin{cases} \dfrac{3\frac{1}{R^2} - 6\frac{1}{R} + 19}{\frac{1}{R^3} - 3\frac{1}{R} + 18} & \text{if } R < 1 \\ 1 + (\dfrac{3}{2}\dfrac{1}{R} + 1)(\dfrac{1}{R} - 1)^3 & \text{if } R \geq 1 \end{cases}, \quad \text{where } R = \frac{\nabla W^C}{\nabla W^{V4}}.$$

B. Adjoint based correction

1. *Linear correction:*

Despite some differences, the usual adjoint based approaches relies on the following implicit correction:[1–6] given an analytical problem defined as

$$\Psi(W) = 0,$$

a numerical discretization of the problem is determined (using here finite volume approach)

$$\Psi_h(W_h) = 0.$$

From the numerical solution an output functional $j_h(W_h)$ is computed. Assuming the numerical solution W_h is close to the analytical solution W, the functional j can be linearize as

$$j_h(\Pi_h W) \approx j_h(W_h) + J \cdot (\Pi_h W - W_h),$$

where $\Pi_h W$ stands for the projection of the exact solution on the mesh and $J = \frac{\partial j}{\partial W}$ is the gradient of j with respect to the solution. The numerical equation can also be linearized as in the Newton's method as

$$\Psi_h(\Pi_h W) \approx \Psi_h(W_h) + A \cdot (\Pi_h W - W_h),$$

where $A = \frac{\partial \Psi_h}{\partial W}$ is the jacobian of Ψ_h. Put together we obtain

$$j_h(\Pi_h W) \approx j_h(W_h) + W^* \cdot \Psi_h(\Pi_h W),$$

where $W^* = J \cdot A^{-1} = A^{-T} \cdot J$ is the adjoint of the problem. This formulation is convenient for mesh adaptation as W^* can be precomputed separately and $\Psi(\Pi_h W)$ can be estimated with local interpolation errors. However, we can see that intrinsically this is equivalent to compute a corrected solution as

$$\widetilde{W} = W_h + A^{-1}\Psi_h(\Pi_h W)$$

and then compute the functional j

$$j_h(\widetilde{W}) \approx j_h(W_h) + J \cdot (\widetilde{W} - W_h),$$

so that the correction on itself is linear.

Similarly, error transport equations are based on the linearization of the given problem:[16–19] a linear correction problem is deduced from the initial problem and thus provide a linear correction of the solution. The linearization of the problem is thus a determining factor of the quality of the corrected solution.[20]

2. Nonlinear correction:

This is why we propose a nonlinear approach for finite volume solvers based on the fact that

$$\Psi_h(\Pi_h W) = \varepsilon_h \neq 0$$

and

$$\Psi(W_h) = \varepsilon \neq 0.$$

In the finite volume approach, the residual is defined as the integral of the continuous residual of a reconstructed solution over a cell or equivalently for inviscid flow as the integral of the fluxes through the boundary of the cell. In that sense, for a given mesh an infinite family of continuous fields $\widehat{W}$ verifies $\Psi_h(\widehat{W}) = 0$, the reconstructed continuous field based on the discrete solution is one of them.

However, if we consider finer meshes, in particular the n^{th} subdivisions of the initial mesh, the only field verifying $\Psi_{h/n}(\widehat{W}) = 0$ for any submesh is the solution of the continuous problem: $\widehat{W} = W$. Thus, the residual of the discrete solution interpolated on a finer mesh $\Psi_{h/2}(I_{h/2}W_h) = \varepsilon_{h/2}$ gives us a usefull indication of the quality of the solution.

We propose the following approach, inspired by multigrid methods, to compute a corrected solution $\widetilde{W}_h$ from the residual $\Psi_{h/2}(I_{h/2}W_h)$. The mesh is initially divided by two (or a power of two) and the solution W_h is linearly interpolated on it. The residual is then computed on the finer mesh and accumulated back on the initial mesh.

The linear accumulation process can be seen as an inverse operation of the linear interpolation: The interpolated value of a vertex of a finer mesh is a linear combination of the values of some vertices of the coarse mesh. In the accumulation process, the residual of this vertex of the fine mesh will be added to the contribution of the vertices of the coarse mesh weighted by the same coefficients. This forms a source term

$$S_h = I_{h/2\to h}\left(\Psi_{h/2}\left(I_{h\to h/2}(W_h)\right)\right),$$

which is then introduced in the equation to compute the correction as

$$\Psi_h(\widetilde{W}_h) = S_h.$$

The corrected solution $\widetilde{W}_h$ is computed performing a few iterations with the same solver, with the source term S_h, which nonlinearly propagates the errors. This approach automatically encompasses all features of the flow solver (order, flux reconstruction, limiters, ...) both in the flux computation and the resolution. All of the operations involved here are local, so that we don't actually need to generate the actual $h/2$ mesh, which would be far too expensive. Instead, we extract for each point its vicinity mesh and divide it by two to compute and accumulate the source terms fluxes. This approach yields no memory over cost and can be done efficiently in parallel. Finer subdivision ($h/4$, $h/8$, ...) of the initial mesh can also be used to improve predictions. The overall procedure is thus:

- Compute the solution on the initial mesh: W_h
- Linearly interpolate the solution on the $h/2$ mesh: $\Pi_{h/2}W_h$
- Compute the $h/2$ residual: $\Psi_{h/2}(I_{h/2}W_h) = \varepsilon_{h/2}$
- Accumulate the $h/2$ residual on the h mesh: $S_h = A_h^{h/2}(\varepsilon_{h/2})$
- Perform a few iterations of the flow solver with S_h as source term: $\Psi_h(\widetilde{W}_h) = S_h$.

Numerical Implementation: Dividing the element size by two multiplies the number of elements by 4 in 2D and 8 in 3D, it is thus particularly expensive to fully generate the submeshes for large practical cases. As the residual computation and accumulation is a local operation, it can be done at a local level, by generating virtually the submeshes of the ball around each vertex. That way no additional memory cost is implied and the process can straightforwardly done in parallel. Similarly, multiple computations of the same element can be avoided with a careful ordering detailed here after.

Figure 2 shows an example of 2D mesh and it's subdivision by two. Starting with the solution on the coarse mesh (circle dots and dark lines), the solution is interpolated on the divided mesh (grey lines and square dots). The residual is computed on the fine mesh and accumulated on the coarse mesh (weighted by the barycentric coordinates).

A careful analysis of the numerical discretization presented in Section II.A shows that the computation of the residual of a given vertex requires the values of the vertices of its third order ball (neighbours of the neighbours of its neighbours). Similarly, to compute the accumulated source term of a given vertex, we only need the vertices of the fine mesh encompassed by the first oder ball of the given vertex in the coarse mesh. This is why, we can generate separately the subdivided mesh of the second order ball for each vertex of the coarse mesh to compute its source term.

However, it can be reduced to the subdivision of the first order ball if we consider carefully the different terms involved in the computation of the residual. In our vertex centered finite volume solver **Wolf**, the advection fluxes are computed and summed for each edge, while the viscous terms are computed for each triangle. We can thus compute them separately in any order as explained below.

We can see in Figure 2 that the residual of the fine node C contributes through the accumulation process to the source term of coarse node A and B by a factor 1/2. The residual of node D contributes to the source term of node A but not of node B (barycentric coordinate 0), however, its solution is required to compute the residual of node C and thus contributes indirectly to the source term of B. Finally, node E directly contributes to the source term of A, but it belongs to an upwind element of the edge DC, so that it contributes to the residual of C and the source term of B.

If we want to compute the source term of node B, the residual of node D does not need to be computed, and the residual of node C which contributes to both source terms of A and B can be computed only once. To do so, we can notice that the advection term in the residual of C is the sum of the fluxes across each edge connecting C to one of its neighbours, *i.e.* here the edges CA, CD, CB, ... As the residual of C is added to the source terms of nodes A and B in the accumulation, we can compute and add separately the contributions of edges CA and BC.

To this end, we generate the local sub-mesh as shown in Figure 3. The grey part of the sub-mesh is not needed and not generated at all, the black part of the sub-mesh is generated for the interpolation of the nodal values and the fluxes are computed only on the red edges and triangles encompassed. That way, only a half of the residual of node C is computed and accumulated on both nodes A and B and the other half of the residual is computed in the vicinity of A. Additionally, the central sub-triangle contribution is computed in the vicinity of the node with the smallest ID in the coarse triangle. Each contribution of each fine edge and triangle are thus computed only once which reduces computational costs. It also ensures that each edge has an upwind and downwind triangles for second order extrapolations in the HLLC flux solver (see Section II.A.4).

III. Numerical Results

In the sequel, given the analytical solution of a problem W and the numerical solution W_h, we split the approximation error $W - W_h$ into the interpolation error and the implicit error as

$$W - W_h = \underbrace{W - \Pi_h W}_{\text{Interpolation Error}} + \underbrace{\Pi_h W - W_H}_{\text{Implicit Error}}.$$

The interpolation error is entirely related to the discretization of the solution and can only be improved by a refinement of the mesh. Here, we are interested in the implicit error as we want to improve the discrete value of our solution. We can thus compute analytically the exact implicit error:

$$\mathcal{E}_{Exa}^{Imp} = \sqrt{\int_0^1 \|\Pi_h W - W_h\|^2 \, \mathrm{d}\Omega},$$

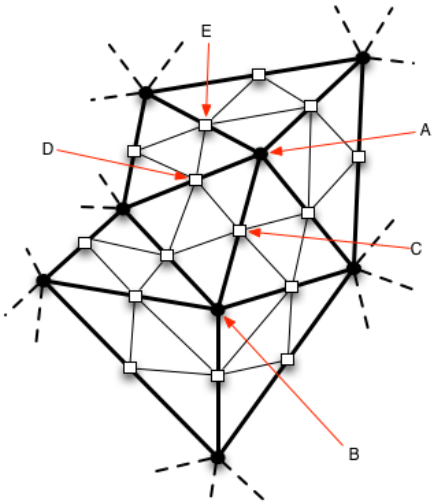

Figure 2. Mesh Subdivision

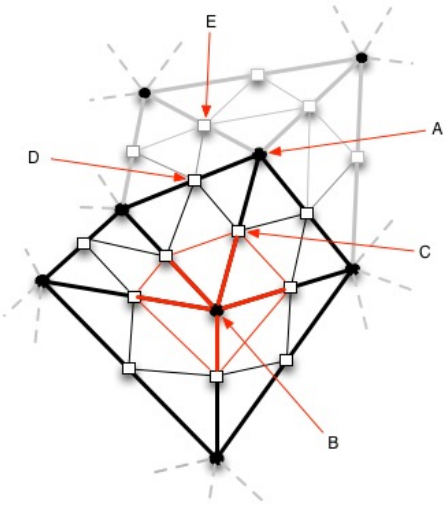

Figure 3. Mesh used for the computation of the source term for node B

the estimated implicit error:

$$\mathcal{E}_{Est}^{Imp} = \sqrt{\int_0^1 \|\widetilde{W_h} - W_h\|^2 \, \mathrm{d}\Omega},$$

as they can be expressed element-wise as polynomial expressions. Similarly, we define a function j computed from u, and we define the exact functional error:

$$\mathcal{E}_{Exa}^{Fun} = |j(W_h) - j(\Pi_h W)|,$$

the estimated functional error:

$$\mathcal{E}_{Est}^{Fun} = |j(W_h) - j(\widetilde{W_h})|,$$

and the corrected functional error:

$$\mathcal{E}_{Cor}^{Fun} = |j(\widetilde{W_h}) - j(\Pi_h W)|.$$

Note that the interpolation error is not taken into account here as, we compare the numerical functional $j(W_h)$ to $j(\Pi_h W)$ which is an approximation of the exact $j(W)$. However this interpolation error can be taken into account in another means. In the sequel, we will use a finer solution as estimation of the exact solution.

A. 2D Laminar NACA0012

We first validate the present corrector in an adaptive context on a NACA0012 geometry, at a Reynolds number of 500 with a purely laminar viscous flow. The Mach number is fixed to $M = 0.1$ and the angle of attack to $\alpha = 3°$. Adapted meshes are generated iteratively with **feflo.a**[21,22] using a goal oriented error estimate on the drag coefficient.[6] An example of the resulting meshes and solutions is shown in Figures 4 and 5.

Figure 8 shows the convergence of the drag computed for both the initial (red) and corrected (blue) solution. We can see that corrector moderately improves the solution and the computation of the drag coefficient. The corrected solution on a 5×10^3 nodes grid provides the same result as the solution on a 8×10^3 nodes grid. The moderate improvement is due to, the mesh adaptation context that introduces a side effect: an adapted mesh with four times the number of vertices of a given adapted mesh is not its subdivision by two. Indeed, it has been further improved by the adaptation and provides a better solution than a subdivision. We will try to quantify this effect with the next test case.

Still, the primal and corrected solutions converge to the same value, so that the difference between both of them gives and accurate estimation of the remaining numerical error. This second property is particularly important in both adaptative and fixed grids context as it informs on how confident we can be in a given solution. It also indicates if the grid convergence study should be continued or not.

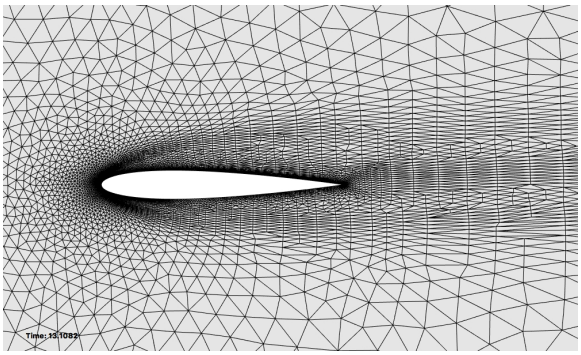

Figure 4. Adapted mesh generated using goal-oriented error estimate on the drag for NACA0012 at $Re = 500$, $M = 0.1$ and $\alpha = 3°$: 8009 vertices.

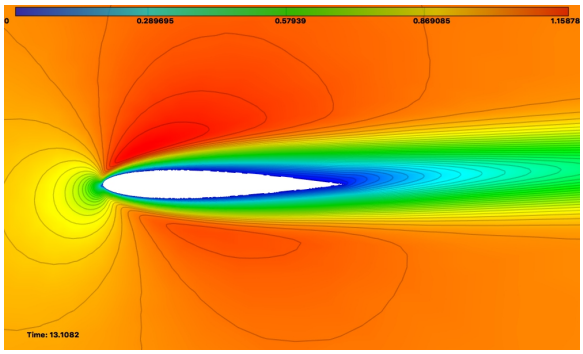

Figure 5. Velocity contours around a NACA0012 at $Re = 500$, $M = 0.1$ and $\alpha = 3°$.

B. 2D Turbulent NACA0012

The corrector is then validated on a turbulent case with the same geometry, at a Reynolds number of 5×10^5 with the Sapalart-Allmaras turbulence model. The Mach number is fixed to $M = 0.7$ and the angle of attack to $\alpha = 1°$. Adapted meshes are generated iteratively with `feflo.a` using a hessian-based error estimate based on the Mach number field. An example of the resulting meshes and solutions is shown in Figures 6 and 7. Additionally, we generate for each mesh its $h/2$ subdivision and compute the solution on it for comparison.

Figures 9 and 10 show the convergence of the drag computed with a first and second order scheme for both the initial (red) and corrected (black) solution. The blue curve represents for each mesh the drag computed on its $h/2$ mesh subdivision which actual complexity is thus four times larger. Both curves exhibits the same trend as the previous laminar example, but it is now possible to quantify how the mesh adaptation improves the solution on a mesh with four times more vertices compared to the actual $h/2$-mesh subdivison. This dims the apparent efficiency of the non-linear corrector. In particular we can see that the corrector is very efficient with a first order scheme as it yields the same solution as the $h/2$-mesh. It is slightly less efficient with a second order scheme but still satisfying.

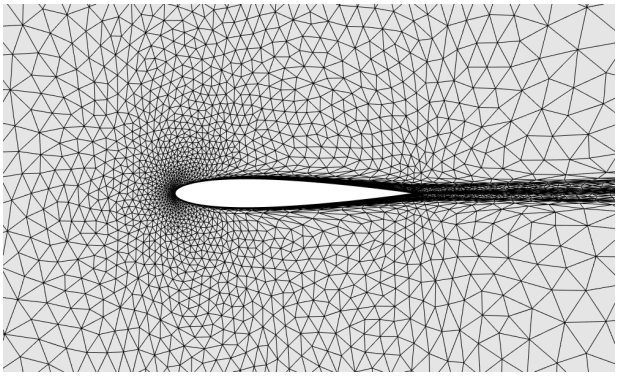

Figure 6. Adapted mesh generated using hessian-based error estimate on the Mach number field for NACA0012 at $Re = 5 \times 10^5$ $M = 0.7$ and $\alpha = 1°$: 5394 vertices .

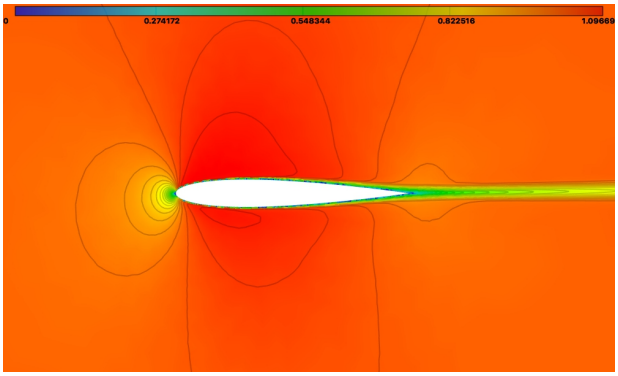

Figure 7. Velocity contours around a NACA0012 at $Re = 5 \times 10^5$, $M = 0.7$ and $\alpha = 1°$.

C. 3D inviscid supersonic case: Sonic Boom Prediction Workshop

The last example is a supersonic case on a realistic aircraft configuration. The considered geometry was proposed for the 2^{nd} AIAA Sonic-Boom Prediction Workshop which aims at minimizing the noise produced by supersonic aircrafts on the ground. Different geometries, meshes and configurations were proposed, we focus here on the C25D flow through Euler case, see Figures 11 and 12. In order to properly lead the shape optimisation process, the noise signature under the aircraft has to be accurate, this is why a grid convergence

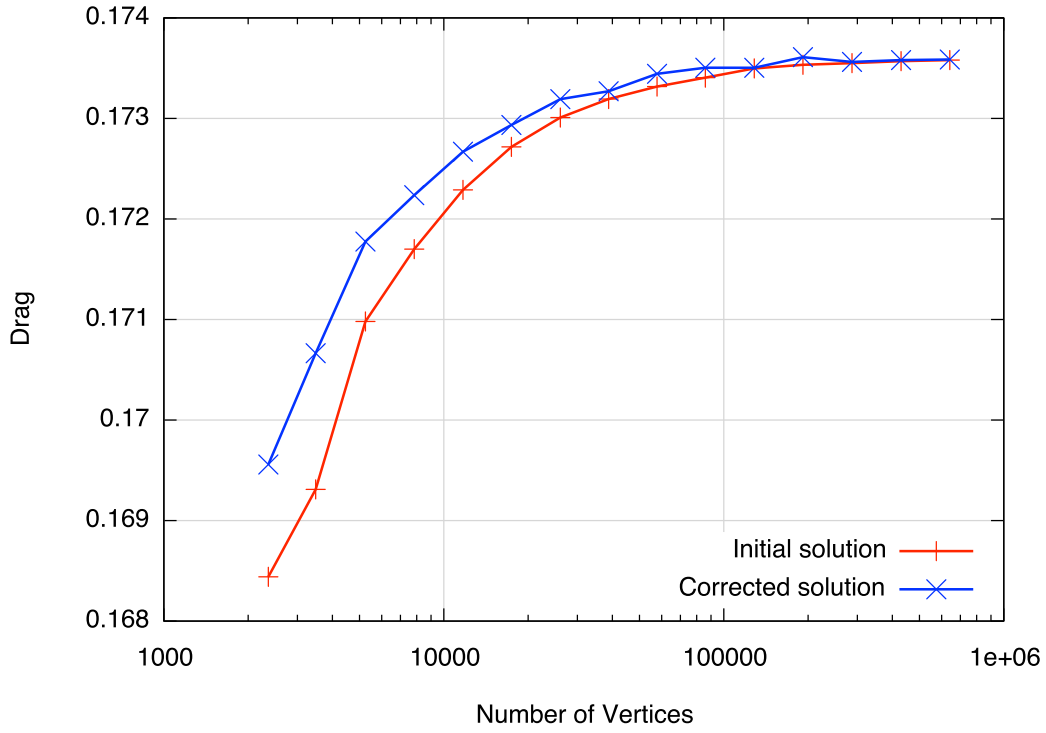

Figure 8. Convergence of the drag on NACA0012 for initial solution (red) and corrected solution (blue) at Reynolds number of 500**.**

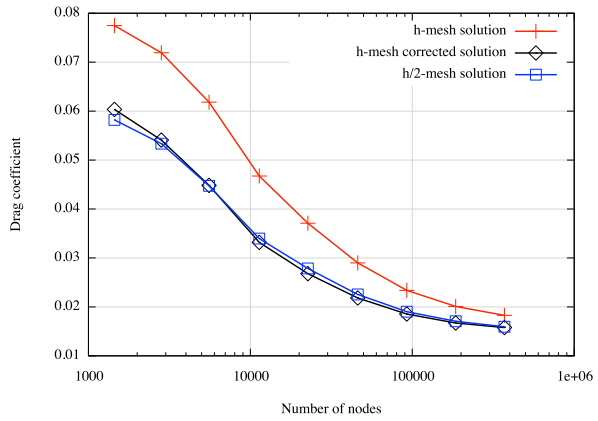

Figure 9. Convergence of the drag with a first order scheme on NACA0012 for the initial solution (red), the corrected solution (black) and the solution on the mesh divided by two (blue) at Reynolds number of 5×10^5**.**

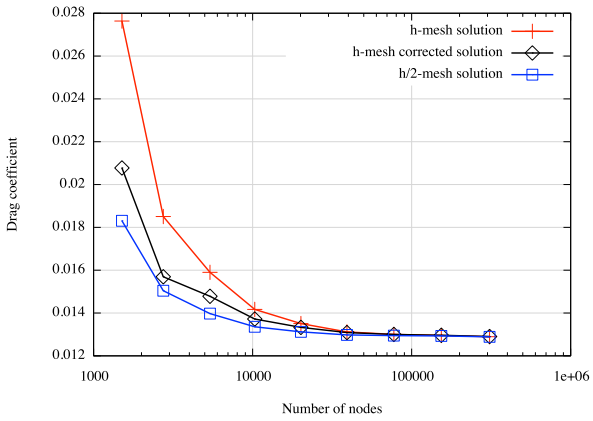

Figure 10. Convergence of the drag with a second order scheme on NACA0012 for the initial solution (red), the corrected solution (black) and the solution on the mesh divided by two (blue) at Reynolds number of 5×10^5**.**

study is required to determine the numerical error. To this end, mesh adaptation has proven to be very efficient[23] and we expect the corrector to provide a second pertinent indication on the convergence of the solution.

Figures 13, 15 and 17 show the pressure signatures (pressure difference with the free stream pressure normalized) under the plane computed on tailored grids provided for the workshop. Error bars represent the estimated error of each nodal value computed with the nonlinear corrector. We can see that the corrector indicates that the error level remains relatively high, even on the 13 million grid, which has been confirmed

by the workshop: the grid convergence is not achieved. This is visible on the leading shock which is still relatively smooth and the shocks in the aft part of the signature ($65 < X < 80$) which are still forming. Note that the actual subdivision by two of this 13 millions nodes mesh would have yielded a 104 millions nodes mesh.

Figures 14, 16 and 18 shows the same computations using goal-oriented mesh adaptation which have been generated to optimally capture the pressure signature.[6] We can se that the error bars are much smaller and tend to reduce during the adaptation process. This is especially visible on the mid part of the signature ($50 < X < 60$). It is even more enlightening to compare the error levels predicted by the corrector on the tailored grid composed of 13 million nodes in the mid part of the signature to the actual difference between the solutions on the tailored grid and adapted grid composed of 5.9 million nodes. The error predicted by the corrector on tailored grid matches the difference between the solution on the tailored grid (which is not converged) and the solution on the adapted grid (which is converged in this region).

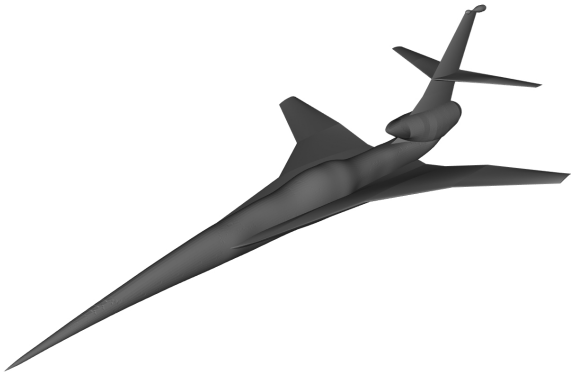

Figure 11. C25D Geometry of the 2^{nd} Sonic-Boom Prediction Workshop.

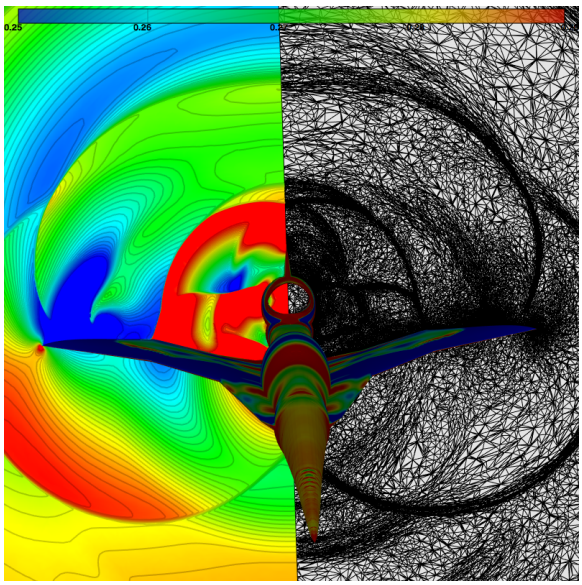

Figure 12. Cut in the volume of the goal-oriented adapted mesh and solution on the C25D geometry.

D. 3D ONERA M6 Wing turbulent Case

We show here the improvements this method can provide on fixed grids in 3D. We choose to use here the ONERA M6 Wing configuration at Mach number $M = 0.1$, angle of attack $\alpha = 3.06°$ and Reynolds number of $Re = 11 \times 10^6$. A series of four grids were generated using **AFLR**,[24,25] with a dimensionless wall spacing based on the Reynolds number of $y^+ = 3,\ 2,\ 1,\ 0.5$. The coarsest mesh is shown in Figure 20.

We can see in Figure 19 that here, the corrector significantly improves the drag prediction, it yields on the 5×10^5 nodes mesh the same drag as the solution without correction on the 3×10^6 nodes mesh. The corrector is even more efficient when the error is not optimally reduced between two grids. We also have a good estimation of the quality of our last solution, without the need of any finer mesh: despite the fine resolution of the boundary layer, we can be pretty confident on the fact that the solution is not converged yet. This is mainly due to the fact that the wake of the wing is poorly discretized, as can be seen in Figure 20. This pollutes the computation of the drag.

IV. Conclusion

We have seen that the nonlinear correction for inviscid, laminar and RANS flow solutions already shows promising results. It improves the overall solution and output functionals, and thus provides a reliable estimation of the numerical error. As it does not requires to effectively generate a finer mesh, it is relatively cheap to use and can provide a pertinent error estimation even on a single grid. It is particularly efficient

on fixed grids and can be used to asses the quality of the mesh refinement that has been done. In a mesh adaptation context its apparent efficiency is reduced as the mesh adaptation process efficiently reduces the error. Though, it is still a useful tool to asses the convergence of the overall process. However, the transonic NACA0012 case showed that the correction is less efficient with a second order scheme than with a first order scheme, suggesting that the MUSCL extrapolation requires an additional particular treatment.

References

[1]Fidkowski, K. J. and Darmofal, D. L., "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics," *AIAA Journal*, Vol. 49, No. 4, 2017/11/06 2011, pp. 673–694.

[2]A Venditti, D. and Darmofal, D., "Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow," Vol. 164, 10 2000, pp. 204–227.

[3]Venditti, D. A. and Darmofal, D. L., "Grid adaptation for functional outputs: application to two-dimensional inviscid flows," *J. Comp. Phys.*, Vol. 176, No. 1, 2002, pp. 40–69.

[4]Formaggia, L. and Perotto, S., "Anisotropic error estimates for elliptic problems," *Numer. Math.*, Vol. 94, No. 1, March 2003, pp. 67–92.

[5]Jones, W., Nielsen, E., and Park, M., "Validation of 3D Adjoint Based Error Estimation and Mesh Adaptation for Sonic Boom Reduction," *44th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-2006-1150, Reno, NV, USA, Jan 2006.

[6]Loseille, A., Dervieux, A., and Alauzet, F., "Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations," *J. Comp. Phys.*, Vol. 229, 2010, pp. 2866–2897.

[7]Spalart, P. and Allmaras, S., "A one-equation turbulence model for aerodynamic flows," *30th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-92-0439, Reno, NV, USA, Jan 1992.

[8]Eca, L., Hoekstra, M., Hay, A., and Pelletier, D., "Verification of RANS Solvers with Manufactured Solutions," *Eng. with Comput.*, Vol. 23, No. 4, Oct. 2007.

[9]Catris, S. and Aupoix, B., "Density corrections for turbulence models," *Aerospace Science and Technology*, Vol. 4, 2000, pp. 1–11.

[10]Batten, P., Clarke, N., Lambert, C., and Causon, D. M., "On the choice of wavespeeds for the HLLC Riemann solver," *SIAM J. Sci. Comput.*, Vol. 18, No. 6, 1997, pp. 1553–1570.

[11]Leer, B. V., "Towards the ultimate conservative difference scheme I. The quest of monotonicity," *Lecture notes in physics*, Vol. 18, 1973, pp. 163–168.

[12]Cournède, P.-H., Koobus, B., and Dervieux, A., "Positivity statements for a Mixed-Element-Volume scheme on fixed and moving grids," *European Journal of Computational Mechanics*, Vol. 15, No. 7-8, 2006, pp. 767–798.

[13]Harten, A., Lax, P., and Leer, B. V., "On upstream differencing and godunov-type schemes for hyperbolic conservation laws," *SIAM Revue*, Vol. 25, No. 1, 1983, pp. 35–61.

[14]Piperno, S. and Depeyre, S., "Criteria for the design of limiters yielding efficient high resolution TVD schemes," *Comput. & Fluids*, Vol. 27, No. 2, 1998, pp. 183–197.

[15]Piperno, S., "Schéma TVD d'ordre élevé pour la résolution de l'équation de Burgers," Rapport de recherche 1996-49, CERMICS, 1996.

[16]Derlaga, J. M. and Park, M. A., *Application of Exact Error Transport Equations and Adjoint Error Estimation to AIAA Workshops*, American Institute of Aeronautics and Astronautics, 2017/11/09 2017.

[17]Hay, A. and Visonneau, M., "Error estimation using the error transport equation for finite-volume methods and arbitrary meshes," *International Journal of Computational Fluid Dynamics*, Vol. 20, No. 7, Aug. 2006, pp. 463–479.

[18]Layton, W., Lee, H. K., and Peterson, J., "A defect-correction method for the incompressible Navier-Stokes equations," *Applied Mathematics and Computation*, Vol. 129, No. 1, June 2002, pp. 1–19.

[19]Pierce, N. A. and Giles, M. B., "Adjoint and defect error bounding and correction for functional estimates," *Journal of Computational Physics*, Vol. 200, No. 2, 2004, pp. 769 – 794.

[20]Yan, G. and Ollivier Gooch, C. F., *Accuracy of Discretization Error Estimation by the Error Transport Equation on Unstructured Meshes - Nonlinear Systems of Equations*, American Institute of Aeronautics and Astronautics, 2017/11/09 2015.

[21]Loseille, A. and Löhner, R., "Cavity-Based Operators for Mesh Adaptation," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, AIAA Paper2013-0152, Dallas, TX, USA, Jan 2013.

[22]A. Loseille, F. A. and Menier, V., "Unique cavity-based operator and hierarchical domain partitioning for fast parallel generation of anisotropic meshes," *Computer-Aided Design*, Vol. 85, 2017, pp. 53–67.

[23]Alauzet, F. and Loseille, A., "High Order Sonic Boom Modeling by Adaptive Methods," *J. Comp. Phys.*, Vol. 229, 2010, pp. 561–593.

[24]Marcum, D., "Adaptive unstructured grid generation for viscous flow applications," *AIAA Journal*, Vol. 34, No. 8, 1996, pp. 2440–2443.

[25]Marcum, D., "Unstructured Grid Generation Using Automatic Point Insertion and Local Reconnection," *The Handbook of Grid Generation*, Edited by J.F. Thompson, B. Soni, and N.P. Weatherill, chap. 18, CRC Press, 1998, pp. 1–31.

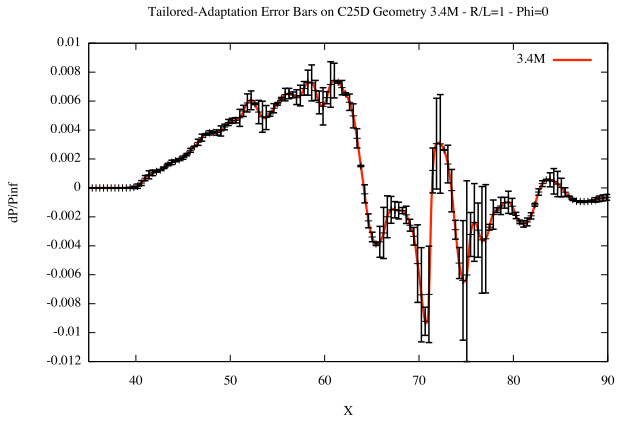

Figure 13. Pressure difference under the plane for tailored grid of 3.4 million vertices (red line) with estimated error provided by nonlinear corrector (black error bars).

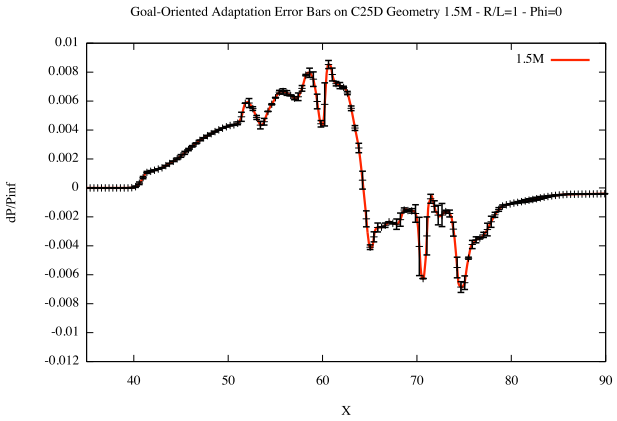

Figure 14. Pressure difference under the plane for goal-oriented adapted grid of 1.5 million vertices (red line) with estimated error provided by nonlinear corrector (black error bars).

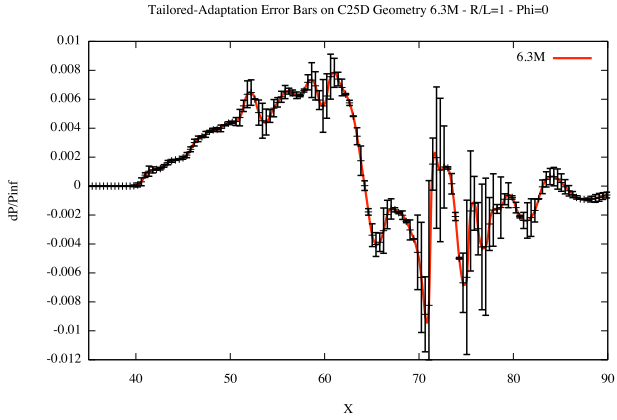

Figure 15. Pressure difference under the plane for tailored grid of 6.3 million vertices (red line) with estimated error provided by nonlinear corrector (black error bars).

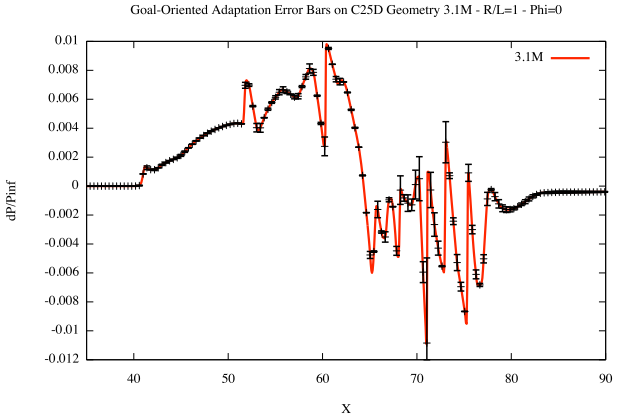

Figure 16. Pressure difference under the plane for goal-oriented adapted grid of 3.1 million vertices (red line) with estimated error provided by nonlinear corrector (black error bars).

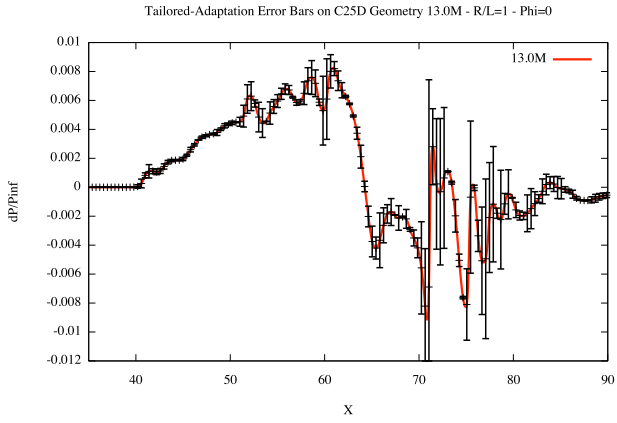

Figure 17. Pressure difference under the plane for tailored grid of 13 million vertices (red line) with estimated error provided by nonlinear corrector (black error bars).

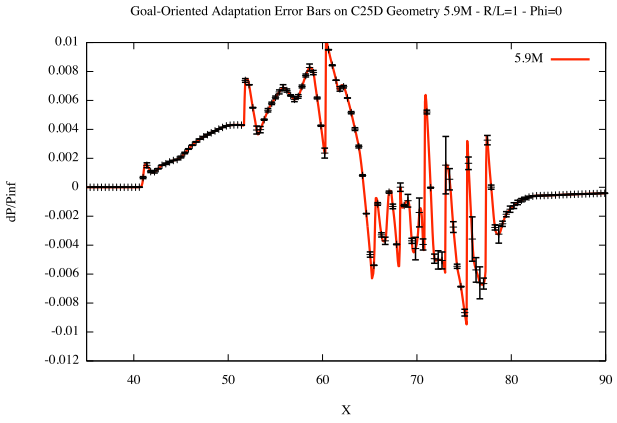

Figure 18. Pressure difference under the plane for goal-oriented adapted grid of 5.9 million vertices (red line) with estimated error provided by nonlinear corrector (black error bars).

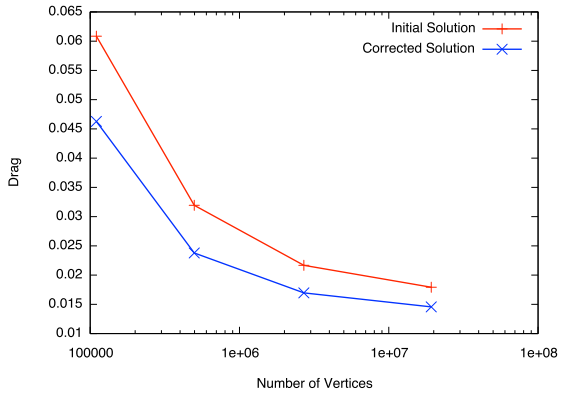

Figure 19. Convergence of the drag on ONERA M6 Wing for initial solution (red) and corrected solution (green) at Reynolds number of 11×10^6 and angle of attack $\alpha = 3.06°$.

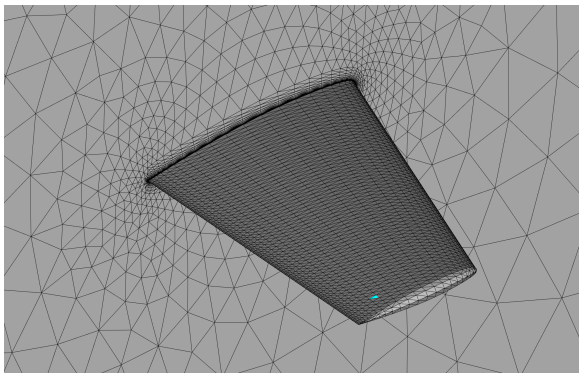

Figure 20. Coarsest M6 Wing mesh generated with a fixed structured boundary layer and frontal approach with AFLR: 110009 vertices.